

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ⁷ :

G06F 9/44

A2

(11) International Publication Number:

WO 00/67118

(43) International Publication Date:

9 November 2000 (09.11.00)

(21) International Application Number: PCT/US00/06690

(22) International Filing Date: 3 May 2000 (03.05.00)

(30) Priority Data:

09/304,419

3 May 1999 (03.05.99)

US

(71) Applicant: NUCOM INTEGRATED TECHNOLOGIES
[US/US]; 2400 Lincoln Avenue, Altadena, CA 91001 (US).(72) Inventors: NUANES, Bruce, P.; 946 North Jackson Street,
Glendale, CA 91207 (US). WALLACE, Greg; 27 Via
Pansa, Rancho Santa Margarita, CA 92688 (US).(74) Agent: YIP, Vincent, K.; McCutchen, Doyle, Brown &
Enersen, LLP, 3 Embarcadero Center, San Francisco, CA
94111 (US).(81) Designated States: CA, JP, European patent (AT, BE, CH, CY,
DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT,
SE).

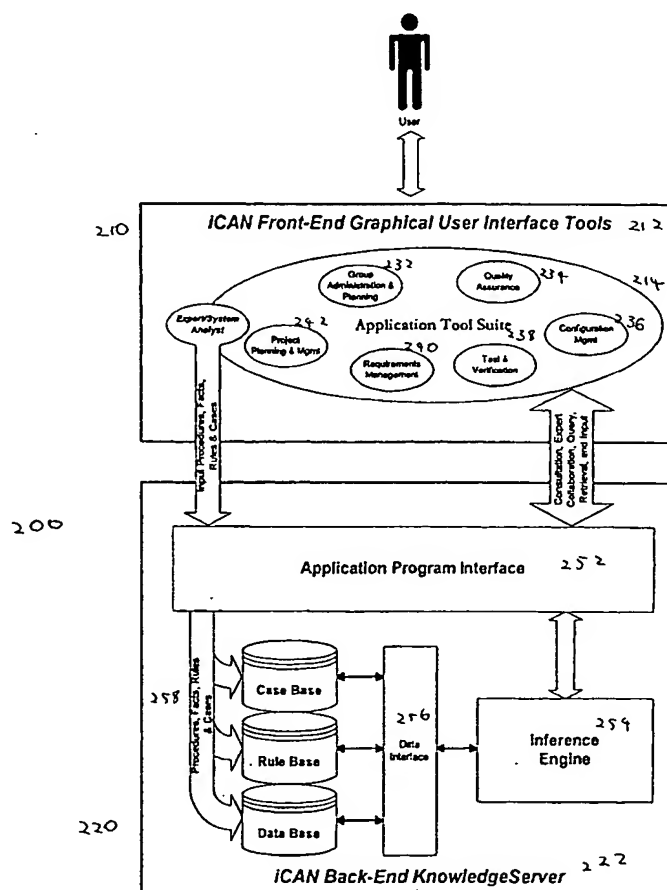
Published

Without international search report and to be republished
upon receipt of that report.

(54) Title: INTELLIGENT COLLABORATION ACROSS NETWORK SYSTEM

(57) Abstract

The present invention describes an integrated advanced systems engineering and complex workflow processes and methodologies, advanced internet technologies, and expert systems. The present invention allows users to: (1) collaborate over a client/server architecture on the evolution of software project plans, system and software requirements, workflow and systems engineering issues, and design details; (2) develop and interact with an evolving knowledge-based expert system within a graphical icon-driven environment; (3) analyze the interactions between a users schedule, goals, and achievement; (4) use real-time information that transforms dynamically based on user-driven input and an embedded expert system, and (5) work more effectively with a secure knowledge-base designed for highly dynamic work setting environments.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
APPLICATION FOR PATENT

5

INTELLIGENT COLLABORATION ACROSS NETWORKS SYSTEM

BACKGROUND OF THE INVENTION

10 This invention relates to the field of software project management, and more particularly, to a method for collecting, organizing, and disseminating engineering data and goals among various users in a software development project.

 In software development project, there have been countless examples that showed expensive consequences of premature implementation without adequate
15 requirements analysis and collaboration on operational concepts. Although a wide array of system development tools are being used in every phase of the conventional life-cycle model, there still remains a deficiency of support mechanisms to manage the global information that is being produced. In addition, there still remains an inadequacy in existing techniques for manipulation and use of the information to
20 support decision making and goal sharing over the internet or intranets. For example, programs developed for NASA, and other aerospace and defense organizations have this attribute since groups as diverse as the mission operations directorate, fault detection, isolation and recovery, safety, software quality, system

development engineers, and programmers must, but often times do not, come together as one unit to understand system level requirements. Current information systems designed to control system and software level requirements, data interface requirements, operational concepts, requirements flow-down, and test and validation
5 processes across all groups, and across a large geographic area, are inadequate because of difficulties in coordination of different information and requirements in the system.

SUMMARY OF THE INVENTION

It is therefore an object of the present invention to provide a method of collecting, processing, and dissemination engineering information and data.

It is also an object of the present invention to provide an efficient
5 software project management method.

It is another object of the present invention to provide a method, apparatus and computer program product of managing software project through a suite of web-based software applications.

It is yet another object of the present invention to provide a method,
10 apparatus and computer program product of managing software project development using an expert system.

The present invention is directed to an integrated advanced software engineering system comprising complex workflow processes and methodologies using expert system analysis. The software engineering system can be distributed
15 among various users located in different locations using advanced intra and/or internet technologies. The preferred embodiment of the present invention allows users to: (1) collaborate over a client/server architecture on the evolution of software project plans, system and software requirements, workflow and systems engineering issues, and design details; (2) develop and interact with at least one evolving
20 knowledge-based expert system within a graphical icon-driven environment; (3)

analyze the interactions between a user's schedule, goals, and achievement; (4) use real-time information that transforms dynamically based on user-driven input and an embedded expert system; and (5) work more effectively with a secure knowledge-base system designed for highly dynamic work setting environments.

5 One of skill in the art will appreciate numerous advantages of the present invention including, for instance (1) overcoming the geographical barrier so that an open architecture and platform independent technology will allow access to resources anywhere on the internet; (2) overcoming communication barriers and providing multi-user accessibility to encourage collaborative software and systems
10 engineering problem solving; (3) maintaining accurate project status so that program, systems and software management personnel can effectively track the progress of their workgroups by using performance measures; (4)an integrated expert system that allows versions of the invention to incorporate expertise developed in the management of previous projects; (5) providing data continuity so
15 that the information can flow between software and systems engineers and program managers can coordinate the work effort in real-time; and (6) supporting software assurance activities by providing technical and objective evidence necessary to support review and acceptable activities.

 Additional objects, features and advantages of various aspects of the
20 present invention will become apparent from the following description of its preferred

embodiments, which description should be taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 shows a top level system operating configuration of a preferred embodiment of the present invention.

Figure 2 illustrates the system level architecture of a preferred
5 embodiment according to the present invention.

Figure 3 shows another embodiment of the system according to the present invention.

Figures 4 a,b,c show a block diagram showing a system control flow design of a preferred embodiment according to the present invention.

10 Figure 5 shows a metric data report of a preferred embodiment according to the present invention.

Figure 6 is a flow chart showing problem and issue report tracking log of a preferred embodiment according to the present invention.

Figure 7 summarizes a process to be modeled of a preferred
15 embodiment according to the present invention.

Figure 8 is an inference engine representation of a preferred embodiment according to the present invention

Figure 9 is a computer system capable of being configured to embody aspects of the invention in accordance with preferred embodiments.

DETAILED DESCRIPTION OF THE DRAWINGS

Figure 1 shows a top level system operating configuration 100 of a preferred embodiment of the present invention. The system operating configuration 100 as shown allows multiple clients 110,112,114,116,118,120,122,124 to connect
5 to an application suite server 130 and to perform various software/systems engineering and project management tasks while maintaining data integrity.

In keeping with terminology common in the art, a client refers to an application or process (either lightweight or heavyweight) that sends a message to a server application or process (again either lightweight or heavywiegth), requesting
10 that the server perform a task (service). The client and/or server may execute on general purpose computing hardware or special-purpose hardware. One of skill in the art will recognize numerous structures of programmed or programmable logic able to be configured to perform functions of client and/or server processes as further described in this specification. Further in keeping with terminology
15 common in the art, reference to hardware configured to provide a client and/or server application may be made by noting "client" or "server" with the understanding that one of skill in the art will appreciate the distinctions.

Further in Fig. 1, the system operating configuration 100 also provides users the capability to remotely access an knowledge-based server 130 over
20 a network. As shown in Figure 1, the application suite server 130 is connected to

various clients 110,112,114,116,118,120,122,124 through different communication channels. For example, local clients 110,112,114,116 are connected to the application suite server 130 through a local ethernet network connection 142. In addition, remote clients 118,124 are connected to the application suite server 130
5 through a modem 126 and a local internet provider 128. Finally, two clients 120,122 are also connected to the application suite server 130 through high speed internet connection 140 such as T1, T2, and T3, etc.

In the preferred embodiment, each of the clients inputs/outputs from/to an end-user; the application suite server 130 interacts with the clients to
10 provide the clients with several levels of functionality. These levels of functionality include providing the clients with display elements such as operator controls, displays, prompts, relational database processing, data retrieval, and expert knowledge-based information.

According to the present invention, the system is implemented in a
15 client/server architecture environment. In the preferred embodiment, the system architecture contains an application server which creates or retrieves database information, advice and consultation provided by an inference engine. The retrieved information, advice and consultation is then supplied to the client(s) for interfacing with users. In addition, the system architecture contains a complete integrated
20 software suite comprising six applications (which will be discussed in detail) that

provide the users with a complete dynamic expert collaborative environment and provide the users access to the tool suite. One of skill in the art will appreciate that the six applications are merely illustrative of various types of applications suitable for obtaining the benefits of an illustrative embodiment of the invention. One of
5 skill will further appreciate that applications may be added or removed from the described embodiments without departing from the scope and spirit of the invention.

In addition, the system architecture allows multiple clients to connect and perform various software and systems engineering and project management tasks while maintaining data integrity. Furthermore, the preferred design supports both the
10 common object request broker architecture ("CORBA") and the common object model ("COM") so that it can provide full client/server support, as well as provide open interfaces and open standards support for extending the platform to a whole world of third party enhancements and growth. The particular client/server architecture is not fundamental to carrying on the features of the invention; one of
15 skill will appreciate that many client/server formats are suitable for use with the invention. Further it is contemplated that the features of the invention will operate with later-developed client-server formats as these technologies evolve and develop.

Figure 2 illustrates the system level architecture of a preferred embodiment according to the present invention. The system level architecture also
20 describes the data and control flow within an inference engine. The system 200 as

shown in Figure 2 comprises two portions: a client portion 210 comprising a front-end graphical user interface 212 for communicating with an user; and a server 220 comprising a back-end knowledge server 222 for processing data and information.

In general, system and requirement engineering involves aspects
5 from system requirement definition, analysis, and management. Requirement definition includes describing a software system with language that clearly defines the needs, operational requirements, and capabilities of the system. It is presented that each requirement is testable in order to prove that the system has implemented the requirement and each is traceable to the design of the software system. Analysis
10 involves verifying that the requirement has been defined sufficiently. In addition, it guarantees that the requirement is not duplicated, generalized, or untestable. It also ensures that the requirement can be fulfilled. Management involves tracking requirement throughout the project life-cycle including handling documentation development, software development, iterative changes to requirements,
15 configuration management base-lining and revision control, test-bed environments, and quality assurance life cycle, etc.

As shown in Figure 2, the front end graphical user interface 212 interfaces with the user for receiving inputs from the user. In the preferred embodiment, the front-end system 210 also comprises an application tool suite 214
20 for performing one or more of the following functions: (1) group administrative and

planning 232; (2) project planning and management 242; (3) systems and requirements engineering 240; (4) test and verification 238; (5) configuration management 236; and (6) software quality assurance 234. Each of these six functions will be discussed in detailed in the following paragraphs.

5 According to the first function (i.e. group administration and planning 232), the system can provide group administration and planning for the users by performing high-level project activities and/or administrative tasks. These include group calendar programming that allows team members to automate the process of setting up meetings and project activities. It also acts as an electronic
10 meeting and notebook support system to increase meeting output, productivity, and the quality of decisions, as well as to provide a discussion area for workgroups to discuss issues, brainstorm, provide timely feedback, and maintain an ongoing record of any discussion that has taken place. The system also performs group project management functions such as to follow-up with meetings. In addition, the system
15 works as a front-end interface for handling electronic mails (i.e. emails) to allow intra and inter-project communications. The system also allows user to setup contact list and to furnish the project with personnel and vendor contacts. Finally, the system also maintain project logs to supply the project team with various logs with which to record pertinent information.

20 According to the second function (i.e. project planning and

management 242), the system is able to handle detailed software project management activities. These project management activities include creation of work breakdown structure, networking various individual tasks, generating time line charts and project tables, allocating resources to various tasks, generating progress
5 reports based on planned schedule, assignment of track tasks; collecting basic project metrics, tracking customer requirements; supporting general project tracking, displaying work standards for practitioners, providing templates for metrics collection, supporting automated computation of function points, and generating metric reports.

10 According to the third function (i.e. requirement management 240), the system performs system and requirement responsibilities involving performing system requirement definition, analysis, and management. In particular, requirement definition includes describing a software system with language that clearly defines the needs, operational requirements, and capabilities of the system.
15 It is presented that each requirement is testable in order to prove that the system has implemented the requirement and each is traceable to the design of the software system. Analysis involves verifying that the requirement has been defined sufficiently. In addition, it guarantees that the requirement is not duplicated, generalized, or untestable. It also ensures that the requirement can be fulfilled.
20 Management involves tracking requirement traceability throughout different project

development cycles including documentation development; software development; iterative changes to requirement throughout life-cycle; configuration management baselining and revision control; testbed environments; and quality assurance life-cycle.

5 According to the fourth function (i.e. test and verification 238), the system of the preferred embodiment supports the development, integration, and formal acceptance of the testing units. These supports include white box testing support (path testing, loop testing, condition testing, data flow testing) and black box testing support. The system not only provides the users with the capability of
10 organizing development, integration, and hardware/software test planning, but also performs test case generation and assists design and coding tools integration. It also supports the system for test case management, regression test management, and test record-keeping mechanism.

 According to the fifth function (i.e. configuration management 236),
15 the system of the preferred embodiment provides software configuration management ("SCM") function. The SCM function is an umbrella activity that may be applied throughout the software engineering process. Because changes can occur at any time, software configuration management activities are developed to identify and control changes. By identifying and controlling changes, it can be ensured that
20 changes are being properly implemented and being reported to others who may have

an interest. A primary goal of software engineering is to improve the ease with which changes can be accommodated and reduce the amount of effort expended when changes must be made. Particularly, the preferred system according to the present invention provides functions including: providing a complete computer
5 software configuration item ("CSCI") identification scheme; supporting all configuration objects including code, text, and graphics; integrating directly with repository; and supporting the change control procedure for all CSCIs. In addition, the system also supports version/release control and provides configuration auditing and reporting features by tracking status of all changes at all documentation levels.

10 According to the sixth function (i.e. quality assurance 234), software quality assurance ("SQA") is defined as conformance to (1) explicitly stated functional and performance requirements, (2) explicitly documented developments standards, and (3) implicit characteristics that are expected of developed software. This definition serves to emphasize three points: (1) software requirements are the
15 foundation from which quality is measured and the lack of conformance to requirements is lack of quality; (2) specified standards define a set of development criteria that guide the manner in which software is engineered where if the criteria are not followed, lack of quality will almost surely result; and (3) there is a set of implicit requirements that often go unmentioned (e.g. the desire for good
20 maintainability). If software conforms to its explicit requirements, but fails to meet

implicit requirements, software quality is suspected. Thus, particularly, the system according to the present invention provides support for technical review (e.g. record keeping, scheduling, and control). It also supports code auditing and establishes support document standards auditing. In addition, the system also provides problem
5 report record keeping such that statistical SQA can be established and managed.

Referring to Figure 2, the back-end knowledge server 222 according to the present invention comprises four major portions. First, the back-end knowledge server 222 includes an application program interface 252 for communicating with the front-end interface 212 and performing application
10 processing. Second, the knowledge server 222 includes a group of knowledge bases 258 (e.g. case base, rule base, and data base) for receiving and storing procedures, facts, rules and cases from the application program interface. Third, the server 222 includes an inference engine 254 for determining whether the set of conditions of the rule base evolution is 'true' given the state of the case base and data base, and
15 assuring that the rule will fire at the appropriate time when responding to a query.

One skilled in the art will recognize that many algorithms used by the inference engine 254 to evaluate the rule base and will choose which rules to fire in the evaluation. Fourth, the back-end knowledge server 222 also includes a data interface 256 for serving as a link between the knowledge bases 258 and the
20 inference engine 254. As it is shown in the figure, the front-end interface 212

provides the back-end interface 222 with input procedures, facts, and rules and cases. In addition, consultation, expert collaboration, query, retrieval, and input are communicated between the front end interface 212 and the back-end interface 222.

Figure 3 shows another embodiment of the system according to the present invention. This figure focuses on the detail design of an inference engine 310. The system as shown similarly comprises three knowledge bases (i.e. data base 322, rule base 324, and case base 326) connected to the inference engine 310 through three data interfaces 328,330,332. In this preferred embodiment, the inference engine 310 receives query requests from an application interface 340.

When an request received is a standard query, the query is processed and data is retrieved from the data base 322. The processed standard query response is then returned to the application interface 340. When the request received is not a standard query, a search is performed on both the rule base 324 and case base 326 knowledge bases to determine whether a potential match is found. If a potential match is found, the query will be processed either by a rule base engine or a case base engine using the facts and goals stored in the system. Finally, a revised solution is provided by the application interface 340 to the inference engine 310 to update the knowledge bases 322,324,326.

Figures 4 a,b,c are a block diagram showing a system control flow design of a preferred embodiment according to the present invention. As shown in

the figures, each of the six rectangular boxes (i.e. group administration and planning 402; project planning and management 404; requirement management 406; test and verification 408; configuration management 410; and quality assurance 412) represents one function (i.e. 232,234,236,238,240,242) in the application tool suite 214 as shown in Figure 2. In the preferred embodiment, different types of data and queries are provided by each of these six functions to a program application. In this preferred embodiment, there are fifteen types of data and queries: test data and result 414 ; requirements 416; problem/issues 418; expenses 420; labor utilization 422; metrics 424; assignments 426; resources 428; tasks 430; work breakdown structure ("WBS") 432; project data and logs 434; contacts lists 436; scheduling 438; emails 440; and electronic notebook 442. These data and queries are then provided to a plurality of program applications such as process test data application groups (i.e. process test support data application 444, process test planning application 446, and process post test data application 448); process requirements tracking and management application 450; process problem and issue management application 452; process metric calculation application 454; process task management application 456; process work breakdown structure and progress generation application 458; process meeting management and project logs application 460; process contacts list and email application 462; process electronic meeting and engineering notebook application 464; process configuration change and version

control management application 466; and process quality assurance auditing and statistics application 468. It should be noted that each of these applications is represented in Figures 4 a,b,c as a circle.

In this preferred embodiment, query results are then returned by the inference engine 470 to the application suite 480 through the data interface management 476. Specifically, the query results are returned to one or any of the six functional boxes 402,404,406,408,410,412. Then, the results are then provided to the users accordingly.

The following provides a brief description of the functions handled by each of these applications.

The process meeting management and project logs application 460 provides functions allowing group team members to follow-up on meetings, review, conversations, and management related issued. In addition, this application 460 also handles different project logging tasks, including preparation of memorandum, meeting minutes, record of discussions, trip reports, and monthly reports, etc.

The process contacts list and email application 462 handles emails transmitted among distributed users. It allows each individual user to setup various email groups as well as send, receive, view, and delete email. In addition, the application 462 also handles the entering, modifying, deleting and querying of contact information.

The process electronic meeting and engineering notebook application 464 handles the interface for supporting free-form writings, sketching, drawings, as well as scanned pictures and tables. It also provides the project team members the capability to respond in real-time to on-line chat and discussions. The application 5 464 also provides the project personnel the capability to perform searches on the textual section of the project notebook, and add, delete and browse any related newsgroups.

The process work breakdown structure and progress generation application 458 establishes a consistent framework for the control and evaluation of 10 the management data. In the software development project, the management data includes statement of work, cost charge numbers, schedules, budgets, plans, earned value, estimate at completion, status reports, and subcontractor integration, etc. This application 458 also provides the user the capability to graphically draw (interactively) a task network showing time-lines, project tables and resource to task 15 allocation. For example, once project data are entered via the network diagram, the application can automatically generate a timeline chart that describes all tasks involved as a function of chronological date. The application 458 can also generate a resource allocation table containing, at a minimum, task start and finish times and work effort to be applied. On the resource to task allocation side, the application 20 allows the users to allocate resources (e.g. money, equipment or people) to specific

tasks, and allows automatic tracking of resource utilization.

The process meeting management and project logs application 460 provides the users the capability to generate reports related to the development projects. These reports include schedule variance reports, metrics overview, project
5 cost data, staffing requirements, risk assessment and abatement, and project and program issues/actions, etc.

The process task management application 456 tracks task assignment by providing the capability to specify each individual assignment, tracking specific task expenses and resources requirements, reporting on outstanding issues and their
10 resolution status, and allowing for the collection of special needs and/or requirements for task completion.

The process metric collection application 454 generates the metric data report as shown in Figure 5 and computes the following values: thousands of lines of code ("KLOC"/pm or Function Points ("FP")/pm; \$/KLOC and \$/FP;
15 KLOC/calendar-month or FP/calendar-month; Defects/KLOC or defects/FP; Effort/defect; and \$/defect. Many other metrics are known in the art and may be used in suitable contexts.

The process problem and issue management application 450 coordinates different engineering effort, issues or problems during the course of the
20 system and software development project. Therefore, it is critical, given the

complexity and the challenging development schedules that software projects introduce, that close coordination is maintained between the contractors and the customers. Particularly, more important than maintaining a close technical and managerial relationship is maintaining an effective corrective action process. Thus, 5 corrections to the system requirements, architecture, software (code or documentation) due to system reviews, analysis, and testing may be required at various stages in the development life-cycle. Problems may be found during peer reviews, formal testing, integration, sell-off or during operation use after delivery. In the preferred embodiment, the system maintains and updates a problem and issue 10 report tracking log. The generation of this problem and issue report tracking log is illustrated in the flow chart as shown in Figure 6.

Referring to Figure 6, problem report inputs/data reported to groups and at meetings are provided to the process problem and issue management application (Step 602) for analysis (Step 604). First, the problem report data is 15 provided to generate track analysis and report(s) and to update case-based scenarios and knowledge base (Step 610). Then, a determination is made to find out whether the problem is required to be reported (Step 606). If a written report is required, a report to the groups and meetings is performed (Step 616) after the report is written (Step 608). If there is no written problem report required, then a determination is 20 performed to determine whether this is a potential issue (Step 612). If this is a

potential issue, a determination is made to find out whether this potential issue is new (Step 614). Finally, a report including the results of all these determinations is generated to groups and meetings in each of these cases (Steps 616,618,620).

In the case of new potential issue, the case-based scenarios and
5 knowledge base will be updated (Step 616). If action(s) are assigned for this new potential issue, a new work action/issue item is issued (Step 626). The new work action issue item is then tracked and stored in the data base (Step 628).

In this preferred embodiment, it should be pointed out that the system provides a clear definition of the issue or problem to be resolved and will contain its
10 origination, personnel involved in issue or problem report, issue/problem to be resolved, severity of issue/problem, status, date of resolution, and responsibility. The system also provides problem and issue management and tracking features by creating, deleting, editing, viewing, and printing issues/problems. According to the present invention, the system provides metric collection such as issue/problems of a
15 given status or those associated to certain personnel.

Referring to Figures 4 a,b,c again, the process task management application 456 also tracks the accounting and budgeting data of the software development project to aid in the prevention of cost overruns. Specifically, the application 456 tracks the provision for a flexible amount of task charge number
20 entries, performs access restrictions of charge numbers to designated employees,

and automatically generates time sheet entries. The application 456 also provides querying capability to track and print hours charged by task, group, or individual. It also reports any charges exceeding allocated budget and performs metrics collection with graphical display of data. In addition, the application 456 also performs

5 administrative function such as authorizing task charges on a per employee basis and/or project basis by analyzing the following accounting items: expense data, authorized expenditure amount, per diem checking, mileage, origination/destination, purpose of expenditure, authorization signature, task charge number and dates of incurred expenses.

10 The process requirements tracking and management application 450 maintains all requirement data within the development project by saving, merging, deleting, creating, viewing, editing, and backing-up requirements data. In addition to maintaining the requirement data, the application 450 also extracts the requirement data from the design and test documentation, as well as from the textual

15 specification. After the requirement data is extracted and processed, the application 450 tracks software/hardware deficiency and associates supplementary data with a requirement describing ambiguity, test status, history of requirement revisions, context of the requirement, author, analytical backup, operational constraints, and author(s), etc. The requirement application 450 also performs requirement

20 traceability function to describe the traceability of a requirement throughout the

development project such as the effect of adding, deleting or modifying a requirement on the entire traceability. Further, the application also provides linkage to software change notices, problem reports, and issues that are driven by requirement additions, modifications, or deletion. Finally, the application 450 acts
5 as an interface with the host-based dictionaries and provides data content representation ("ICDs") and query support to data dictionary and/or repository.

As discussed, the inference engine 470 of the preferred embodiment performs system engineering inferences on the software development project. It is known that as software development become more complex, the methods for
10 developing these products and managing the design process have become more complex. The inference engine 470 used in the preferred embodiment of the present invention employs a methodology focusing on how to systematically define, measure, stabilize, and improve each individual process performance during the software development project. Moreover, the method describes different practices
15 of gathering, developing, and using a variety of information and data to improve user satisfaction and to reduce waste within an integrated product team environment.

Figure 7 summarizes the process to be modeled, and containing the systems engineering facts, procedures and goals.

As shown in Figure 7, the inference engine first defines the process
20 (i.e. procedures, facts, rules, and cases) (Step 710). After the process is defined,

applicable metrics and/or goals are then established (Step 712). Then, the system defines all the performance criteria within the defined process and applicable metrics (Step 714). The inference engine will then be provided with any stabilization factors, if available (Step 716). The process will return to reestablish the applicable metric if improvement is required (Step 718). On the other hand, if improvement is not required, new goals will be defined (Step 720) and the corresponding new procedures and facts will be developed (Step 722) so that new cases can be implemented (Step 724).

The above-mentioned procedure assumes the following systems engineering environment: (1) tasks are performed via integrated product teams, (2) all work is managed using standard program management processes, (3) accountability to the customer is required for day-to-day execution of program business (e.g. quality, quantity, timeliness, and budget), (4) task requirements, schedule, work acceptance, and team composition will require agreements between the team leader and the lead manager, (5) the system will provide configuration auditing and reporting features, and (6) the system will also track status of all changes at all documentation levels.

In general, automated management systems can help to cope with the increase in the number and complexity of events by (1) automating the collection and reporting of events, thereby reducing the load on human operators or programs,

(2) using even correlation techniques to group distinct events, thereby compressing the event stream into a form more easily managed by human operators, (3) mapping groups of events to their underlying causes, thus reducing the time between faults and repairs, and (4) automatically correcting diagnosed problems, thereby
5 minimizing operator intervention.

Event correlation and management techniques are a particularly important method of reducing the number of symptoms in a system which need to be analyzed. The correlation and management techniques should be able to accurately identify and determine the number of discrete problems which need to be
10 rectified. Unless events are correlated, a single problem in a single subsystem could result in multiple, uncoordinated correction actions. This can lead to wasteful resources spent on duplicate efforts and inconsistent corrective actions which result in escalation of problems.

Event correlation and management approaches can be generally
15 grouped into five categories: (1) rule-based reasoning; (2) case-based reasoning; (3) reasoning with generic models; (4) probability networks; and (5) model-based reasoning. In addition, a number of different architectures have been considered to carry out event correlation and management. In order to review these approaches, the following terminology is defined:

20 Knowledge Representation: the format and means for representing

knowledge about the software development system being monitored, such as test data and results, problems/issues, expenses, labor utilization, metrics, etc. Examples of these knowledge are shown in Figures 4 a,b,c. Such knowledge may be stored in knowledge base organized as a hierarchical relational or object-oriented database.

5 Knowledge Acquisition: The methods and means for acquiring the knowledge about the system to be monitored. Ideally, knowledge is obtained during system operation to minimize human resource requirements.

Event Correlation: The methods and means for detecting the occurrence of exceptional events in a complex system and identifying which
10 particular event occurred and where it occurred. The set of events which occur and can be detected in the system over a period of time will be referred to as an "event stream." It will be noted that the location of the event is not necessarily the location where it is observed, because events can propagate across related entities in a system. Although every possible reportable measurement (such as KLOC, pm, and
15 expenses, etc.) could be considered to be an "event", many of these measurements do not contribute to identifying exceptional events in the system. Event correlation takes as input an event stream, detects occurrence of exceptional events, identifies the particular events that have occurred, and reports them as an output.

 Event correlation can take place in both the space and time
20 dimensions. For example, two events whose sources are determined to be in the

same protocol layer in the same network element may be related spatially. However, they may not be correlated if they occur on different days, because they would not be related temporally.

1. Rule-Based Reasoning Methods

5 One approach for correlating events in complex systems involves rule-based reasoning, such as expert systems. Rule-based expert systems generally contain two components: (1) a working memory which represents knowledge of the current state of the system being monitored; and (2) a rule base which contain expert knowledge in the form of “if-then” or “condition-action” rules. The condition part
10 of each rule determines whether the rule can be applied based on the current state of the working memory; the action part of a rule contains a conclusion which can be drawn from the rule when the condition is satisfied.

Rule-based reasoning can proceed in one of two possible modes of operation. In “forward chaining” mode, the working memory is constantly scanned
15 for facts which can be used to satisfy the condition part of each rule. When a condition is found, the rule is executed. Executing a rule means that the working memory is updated based on the conclusion contained in the rule. These newly updated data can be used to satisfy the conditions of other rules, resulting in a “chain reaction” of rule executions.

20 In “backward chaining” mode, the system is presented with a “goal”

working memory datum, which it is asked to either confirm or deny. The system searches for rules whose action part could assert the goal; for each such rule, the condition corresponding to the action is checked against the working memory to see if it is satisfied. The conditions can be satisfied by either finding the appropriate
5 working memory data or by finding other rules whose conditions are satisfied which could assert the desired working memory data.

Rule-based expert system benefit from straightforward knowledge acquisition because the “if-then” format of the rules often mimics the format of expert knowledge. The knowledge base can be incrementally modified because
10 rules can be added or modified easily. However, attempts to automate knowledge acquisition for such systems have produced limited results.

2. Case-Based Reasoning Methods

Case-based reasoning methods and systems involve storing knowledge as a repository of successful cases of solved problems called a case base.
15 When the system is presented with a problem, it searches the case base for similar cases. Once the similar cases are retrieved, various problem solving strategies must be adapted to the case at hand. If the adapted strategy successfully solves the problem, then the newly solved problem can be added to the case base with the adapted solution.

20 One way to more closely match problems with those in the case base

is to use “determinators.” Determinators are a way of narrowing the similarity criteria to attributes of a problem which are relevant to solving the problem. For example, the solution to the problem “file transfer throughput is slow” could be determined by looking at the bandwidth, network load, packet collision rate and
5 packet deferment rate; these would constitute determinators. Parameterized adaptation such as interpolating among solutions to similar problems located in the case base can be used to provide solutions to new problems.

3. Reasoning with Generic Models

Generic models rely on generic algorithms, rather than expert
10 knowledge, to correlate events based on an abstraction of the system architecture and its components. As an example, each event can be normalized to include a list of all possible faults which could have been responsible for the event. This is abstraction of a real event which could carry much more varied information. Then all the various events are collected and the intersection of their sources is determined
15 and output as the diagnosis.

As an example, if events A and B are detected, and it is known that event A could have been caused by problems 1, 2, or 3, and event B could have been caused by problems 2, 4, or 6, then the diagnosis is that problem 2 has occurred because it represents the intersection of the possible sources of events A and B. The
20 complexity of this approach is generally the number of events multiplied by the

number of source faults which could have generated the events.

4. Probability Networks

The various approaches outlined above can be augmented with probability information. For example, a rule of the form "if A then B" can be
5 augmented with a certainty factor: "if A then B with certainty 90%."

The element of a probability network is a proposition, which is a hypothesis about the state of the system being monitored. For example, the hypothesis "node A is faulty" is a proposition. A probability is associated with each proposition, which is its a priori probability of truth. Additionally, probabilities can
10 be assigned to the relationships between propositions. For example, "the truth of proposition A causes the truth of proposition B with probability 90 %." When an event occurs, the probability of the proposition representing the occurrence of that event is updated to 100%, and this change is propagated to other propositions in the network based on the relationships. A diagnosis can be generated by simply listing
15 those propositions having the highest probabilities.

Another approach which can be included in this category is often
referred to as "fuzzy backward reasoning" ("FBR"), based on principles of fuzzy
logic. Fuzzy logic describes uncertain knowledge in terms of subintervals of [0,1].
For example, the likelihood of a problem can be represented as an interval [0,0.4].
20 The certainty (fuzziness) of the problem is given by 0.4. Fuzzy logic, in a manner

similar to Boolean logic, defines operations in terms of intervals. The product of two intervals is their intersection, while the sum is their union.

FBR can be used to model causality among problems and symptoms using a matrix R of fuzziness indicators. For a vector a of problems and a vector b of symptoms, the problem of fuzzy backward reasoning can be defined as
5 computing the problem vector a that solves the equation $b=a*R$, where vector b represents symptoms, a represents problems, and R represents fuzziness indicators.

5. Model-Based Reasoning

Model-based reasoning involves creating a model which represents
10 the underlying system being monitored. One example of a model is a finite state machine ("FSM") for modeling possible states of the system. As messages are observed at any location in the system, the model is used to update the estimate of the current state of the system.

It should be noted that, for the present invention, any one of the
15 above-mentioned methods can be implemented in the inference machines of the present invention to provide consultation, expert collaboration for the application program interface. Thus, a brief discussion on using four of the above-mentioned reasoning methods (i.e. semantic network, frames, case-based and rule-base reasoning techniques) in implementing the inference machine is provided as
20 follows.

In the semantic network approach, relationships between various entities are created and analyzed to describe systems, ideas and problems. Then, by analyzing the events and these stored relationships, inferences can be made.

On the other hand, in the frames approach, data structure is organized
5 in frames. The frames are used to represent the "stereotyped" situations so that each frame contains different kinds of information such as: how to use the information; what is going to happen next; and what to do if expectations are not committed, etc.

As to the case-based reasoning approach, a case is a contextualized piece of knowledge representing an experience. It contains the past lesson that is the
10 content of the case and the context in which the lesson can be used. In the present system, the problem space describes the state of the project or scenario when the case occurred, and the solution space states the derived solution to that problem. Figure 8 is an inference engine representation of a preferred embodiment according to the present invention. As shown in Figure 8, the description of a new problem to
15 be solved is positioned in the problem space. Retrieval identifies the case with the most similar problem description. If necessary, adaptation occurs and a new solution is created. The present invention may use indexing to speed retrieval of data.

As to the rule-based expert system approach, knowledge is
20 represented as facts about the world (i.e. relationships between entities) and rules for

manipulating the facts. This apparent simplicity is complicated by the problems that at any one time more than one rule may apply and that as each rule is applied many more may become applicable. The present system will implement a control structure to decide which rule to apply next and how to chain rules together. In a
5 preferred embodiment, the present system uses a "depth-first forward chaining" control structure as shown in Figure 7. However, an inherent problem in inference engines is in which order to process rules - a decision must be made as to which rule to "fire" first. In the depth-first forward chaining structure, inferences would be made based on the order indicated as shown in Figure 7. Thus, in the present system
10 as shown, G would be inferred as true before it inferred that E was true. If G is a solution, there may not be a need to infer that E was also a potential solution.

In a preferred embodiment, the solutions derived by the inference machine are based on a process called adaptation. The adaptation process looks for prominent differences between the event and case(s) retrieved from either one of the
15 knowledge bases. Then, the inference machine applies the corresponding formulas and rules to the case(s). In the process, the inference machine takes those differences into account when suggesting a final solution. In this preferred system according to the present invention, there are various methods of adaptation such as null adaptation; parameter adjustment; reinstatement; derivational replay; and
20 model-guided repair. In determining which is the most appropriate adaptation

method to derive the solution to a given problem, the preferred embodiment uses an internal rule-based mechanism.

Following are examples showing different features of the present invention.

5 The first example illustrates the null adaptation method. The null adaptation applies the retrieved solution to the current problem (i.e. case) without adapting the solution to the specific problem.. Specifically, this example is directed to solving the problem of the rear door of minivans that swings open during minor collisions and accidents. According to this method, an identical case is located in
10 the case-based knowledge base in the case retrieval process. The case is then retrieved along with its associated "lesson learned" (i.e. solution). Then, the lesson learned is used to solve the current problem.

 The second example illustrates the parameter adjustment method. The parameter adjustment method compares specific parameters of the retrieved
15 case with those of the current case such that the solution can be modified in an appropriate direction. Specifically, this example is directed to solving the problem of preparing a test plan for a flight control system for the new joint strike fighter ("JSF"). In this example, the retrieved case comprises a previously successful test plan that covered a similar flight control system which can be appropriately
20 modified. The retrieved case is then used to satisfy the JSF program's requirements

for testability, verification and validation.

The third example illustrates the instantiation method. The instantiation method is used to instantiate features of an old solution with new features. Specifically, the example is directed to document specific requirements of
5 a constellation of satellites system during its planning stages. In this example, once the first satellite's requirements are documented, each subsequent satellite's requirements can be instantiated from the first satellite's requirements.

The fourth example illustrates the derivational replay method. The derivational replay method is a process of retracing the method used to arrive at an
10 old solution so that a new solution in a new situation can be developed. Specifically, the present example is directed to investigate the issues raised on the use of O-ring in a proposed solid rocket booster ("SRB") design. According to this method, the retrieved case identifies problems discovered on the space shuttle SRBs and the steps taken to solve that problem. Then, an appropriate solution can be
15 derived based on these data.

Finally, the fifth example illustrates the model-guided repair method. The model-guide repair method is a causal model to guide adaptation. Specifically, the example is directed to find a solution on the frequent black out problem reported in south western United States. According to this method, it could be inferred that
20 the problem could be attributed to heat waves in that region because of a causal link

between weather and black outs.

Methods according to the invention may be computer implemented either in whole or in part. Figure 9 depicts a computer system 900 capable of embodying aspects of the invention. Hardware on which clients
5 110,112,114,116,118,120,122 or server 130 execute may be hardware in accordance with the computer system 900. The computer system 900 comprises a microprocessor 910, a memory 920 and an input/output system 930. The memory 920 is capable of being configured to provide a data structure 940 which may contain data manipulated by the computer system 900 when embodying aspects of
10 the invention. Further illustrated is a media drive 970, such as a disk drive, CD-ROM drive, or the like. The media drive 970 may operate with a computer-usable storage medium 975 capable of storing computer-readable program code able to configure the computer system 900 to embody aspects of the invention. The input/output system 930 may also operate with a keyboard 950, a display 960, a
15 pointing device 990, or a data network interface. As illustrated, the computer system 900 is general-purpose computing machinery. As one of skill recognizes, programmed instructions may configure general purpose computing machinery to embody structures capable of performing functions in accordance with aspects of the invention. Special purpose computing machinery comprising, for example, an
20 application specific integrated circuit (ASIC) may also be used.

In an illustrative embodiment of the invention, computer program code configures a computer to embody aspects of the invention. So configured, the computer provides a means for performing functions described above in connection with operation of client and server applications. So configured representations of
5 physical quantities and characteristics are manipulated through a series of operations to achieve aspects of a method, apparatus, and system for intelligent collaboration across networks.

It is to be understood that while the invention has been described above in conjunction with preferred specific embodiments, the description and
10 examples are intended to illustrate and not limit the scope of the invention, which is defined by the scope of the appended claims.

What is claimed is:

1. A management system for managing a software development project comprises a plurality of tasks, said management system comprising:
 - a host server, said host server comprising at least one application
 - 5 program for performing at least one of said tasks in the software development project;
 - and
 - at least one client server connected to said host server, said client server interfacing with a user,
 - wherein said host server further comprises an inference engine for
 - 10 receiving queries from the application program, said inference engine making inferences from said queries and thereby providing said application program with corresponding inferences.
2. The management system according to claim 1, wherein said
- 15 inference engine is a case-based reasoning mechanism.
3. The management system according to claim 1, wherein said inference engine is a rule-based reasoning mechanism.
- 20 4. The management system according to claim 1, wherein said

management system comprises more than one client servers, and wherein at least one of the client servers is connected to said host server through ethernet.

5. The management system according to claim 1, wherein said
5 management system comprises more than one client servers, and wherein at least one of the client servers is connected to said host server through internet.

6. The management system according to claim 1, wherein said
management system comprises more than one client servers, and wherein at least one
10 of the client servers is connected to said host server through high speed communication line.

7. The management system according to claim 1, wherein said
management system comprises more than one application programs, and wherein at
15 least one of the application programs is a group administrative and planning suite for performing high-level project activities and administrative tasks for the software development project.

8. The management system according to claim 1, wherein said
20 management system comprises more than one application programs, and wherein at

least one of the application programs is a project planning and management suite for handling a plurality of activities in the software development project.

9. The management system according to claim 1, wherein said
5 management system comprises more than one application programs, and wherein at least one of the application programs is a systems requirements engineering suite for handling system requirement definitions, analysis and management in the software development project.

10 10. The management system according to claim 1, wherein said management system comprises more than one application programs, and wherein at least one of the application programs is a test and verification suite for supporting development, integration, and formal acceptance in the software project development.

15 11. The management system according to claim 1, wherein said management system comprises more than one application programs, and wherein at least one of the application programs is a configuration management suite for providing software management functions in the software project development.

20 12. The management system according to claim 1, wherein said

management system comprises more than one application programs, and wherein at least one of the application programs is a software quality assurance suite for providing software quality assurance functions in the software project development.

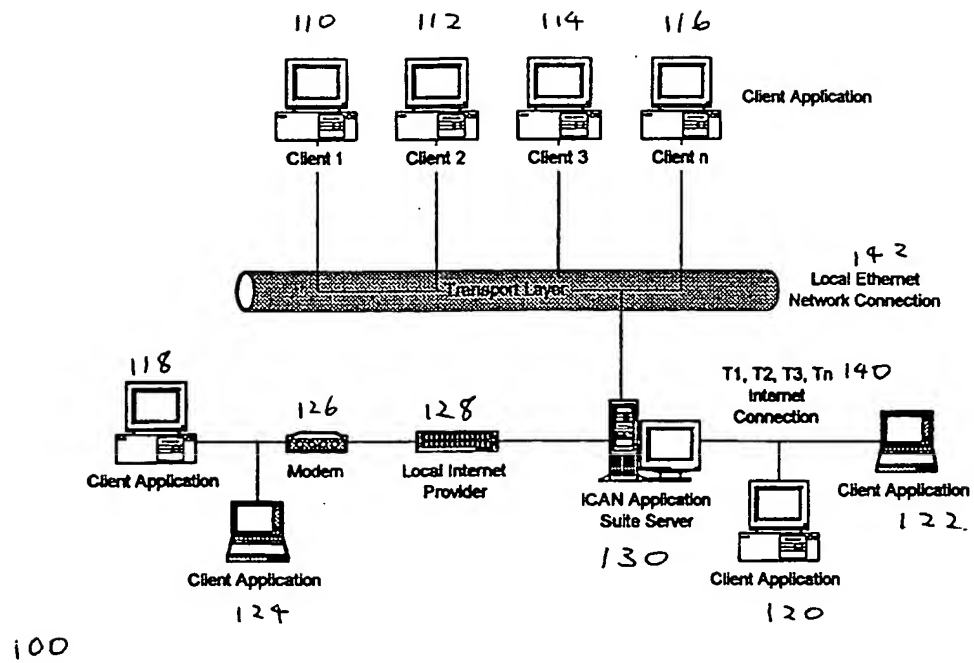
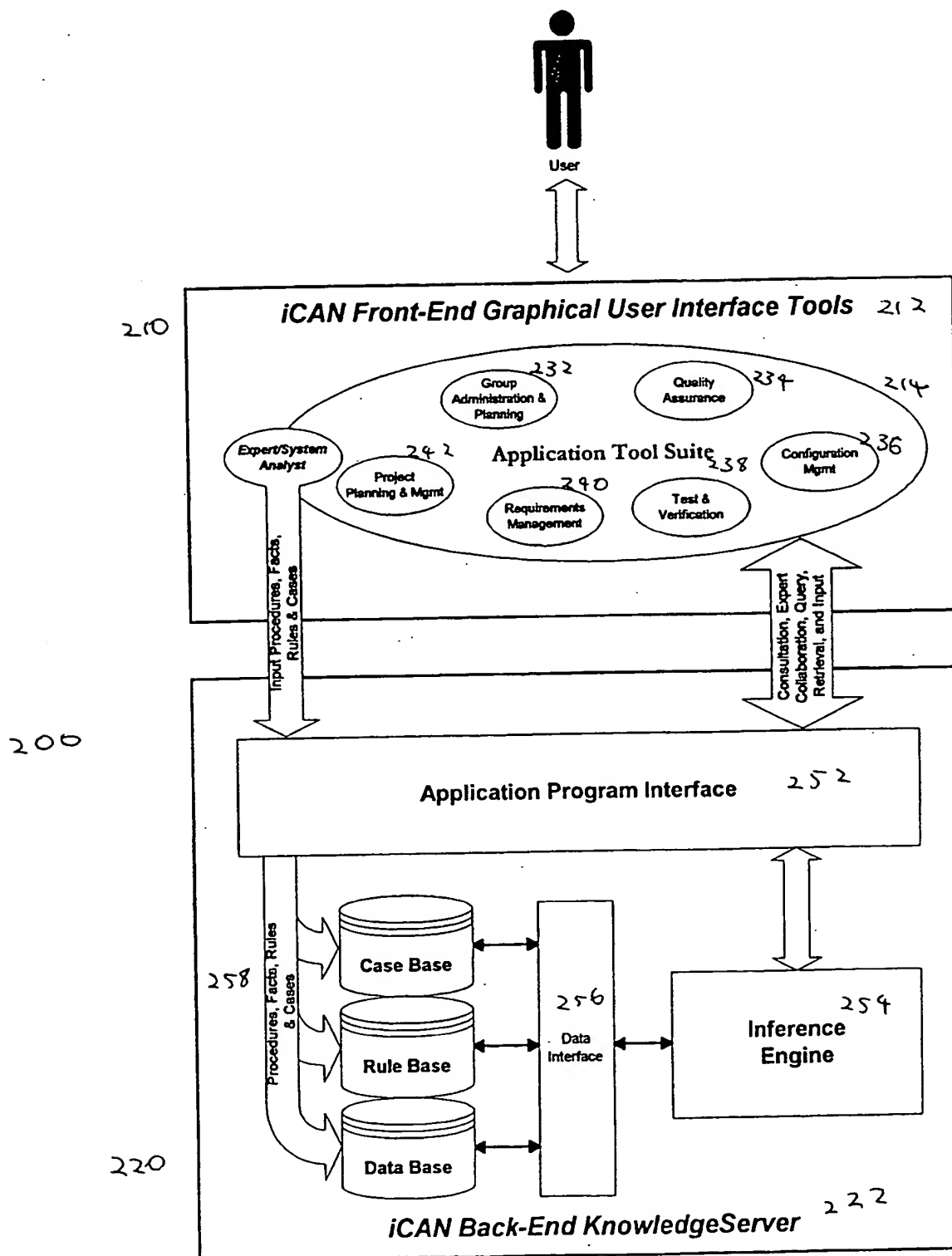


Figure 1



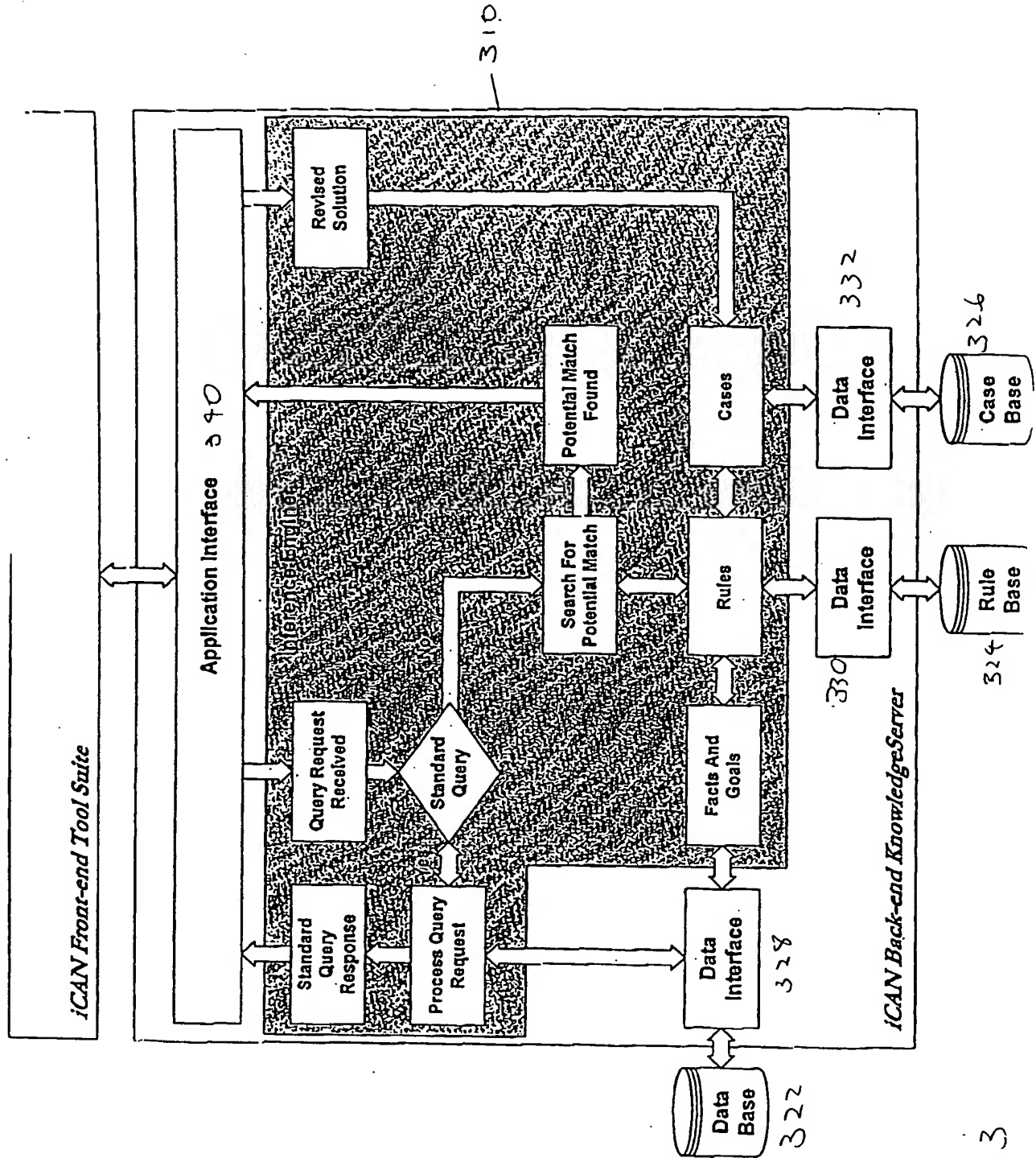


Figure 3

Figure 4a

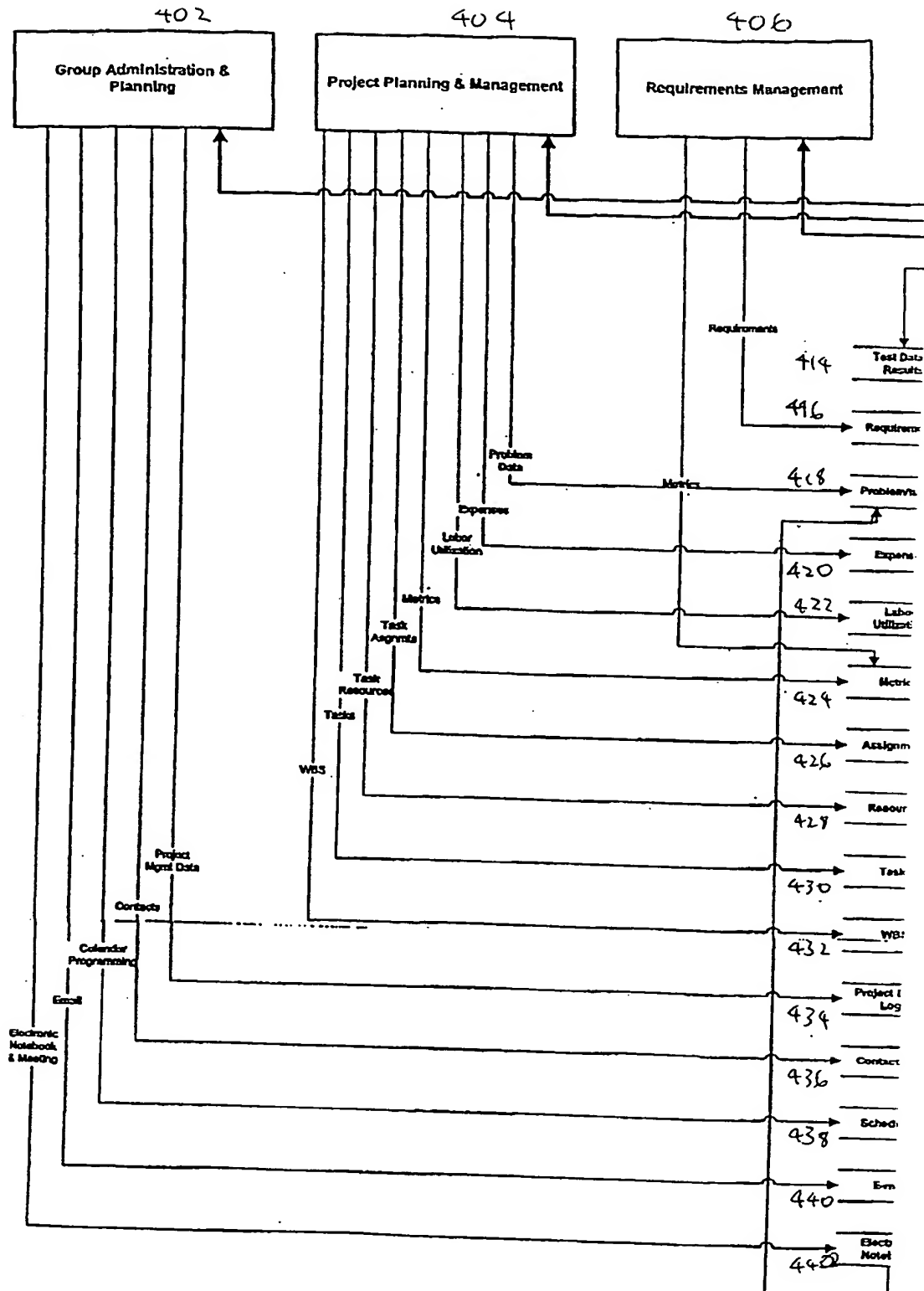


Figure 4b

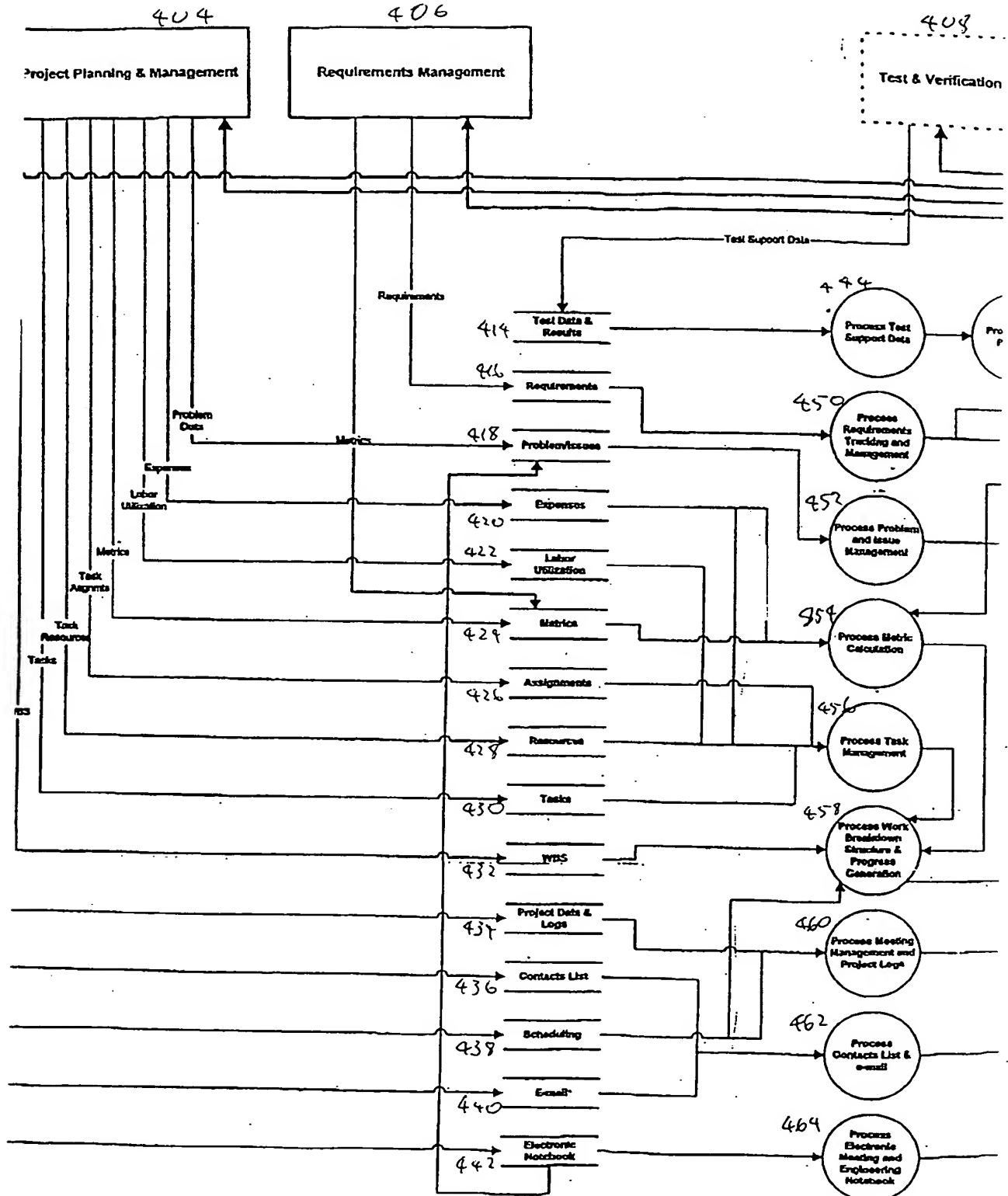
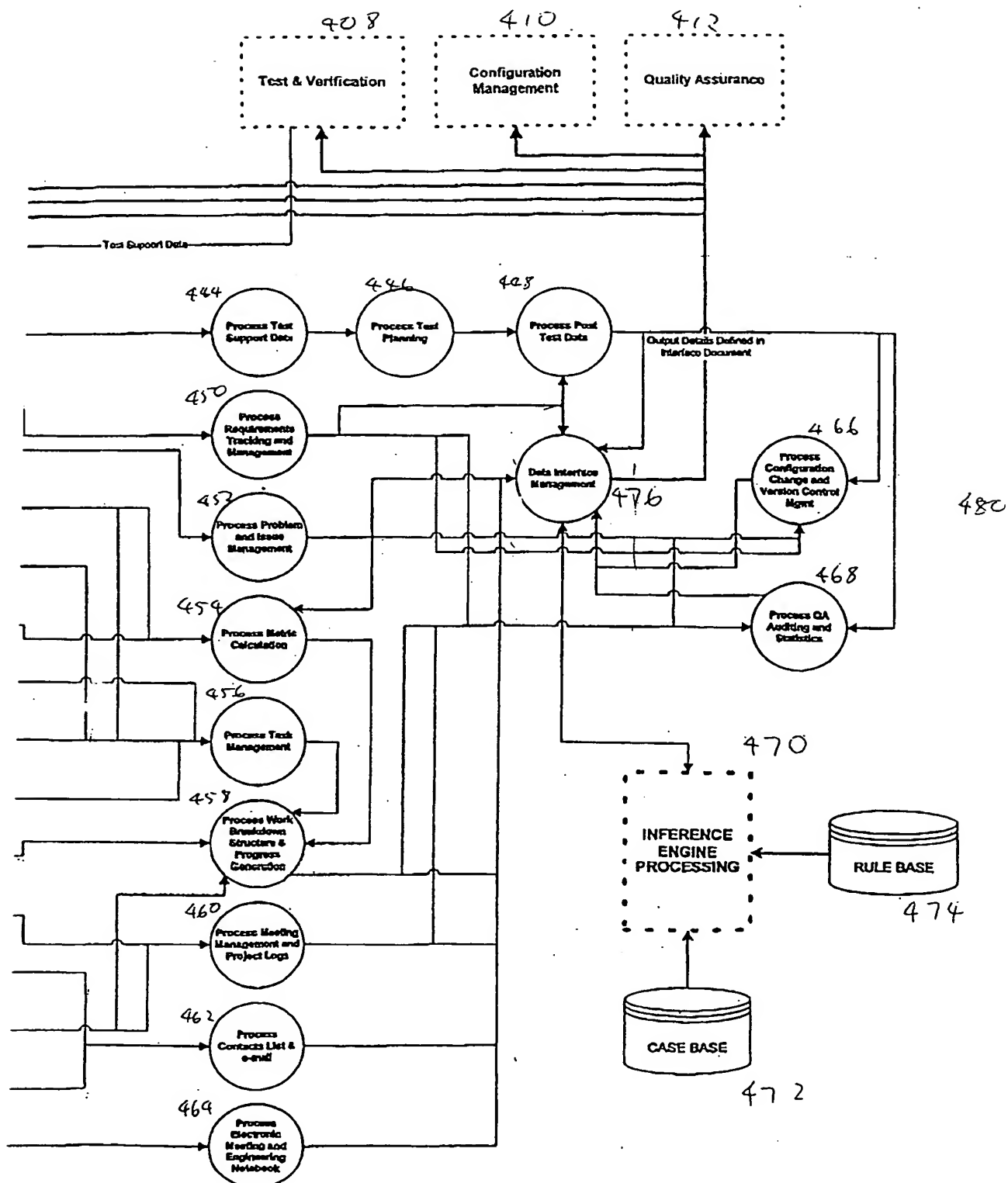


Figure 4c



Cost Data – Input	
Description	Units
Labor cost	\$/person-month
Labor year	hours/year
Data for Metrics Computation	
Description	Units
Project name or identifier	Alphanumeric
Release type	(development – maintenance)
Number of project staff	People
Effort	Person-hours
Effort, computed	Person-months (pm)
Elapsed time to completion	Months
Newly developed output	KLOC or FP
Modified (existing code)	KLOC or FC
Output delivered (total)	KLOC or FP
Number of defects (after release)	Defects

Figure 5

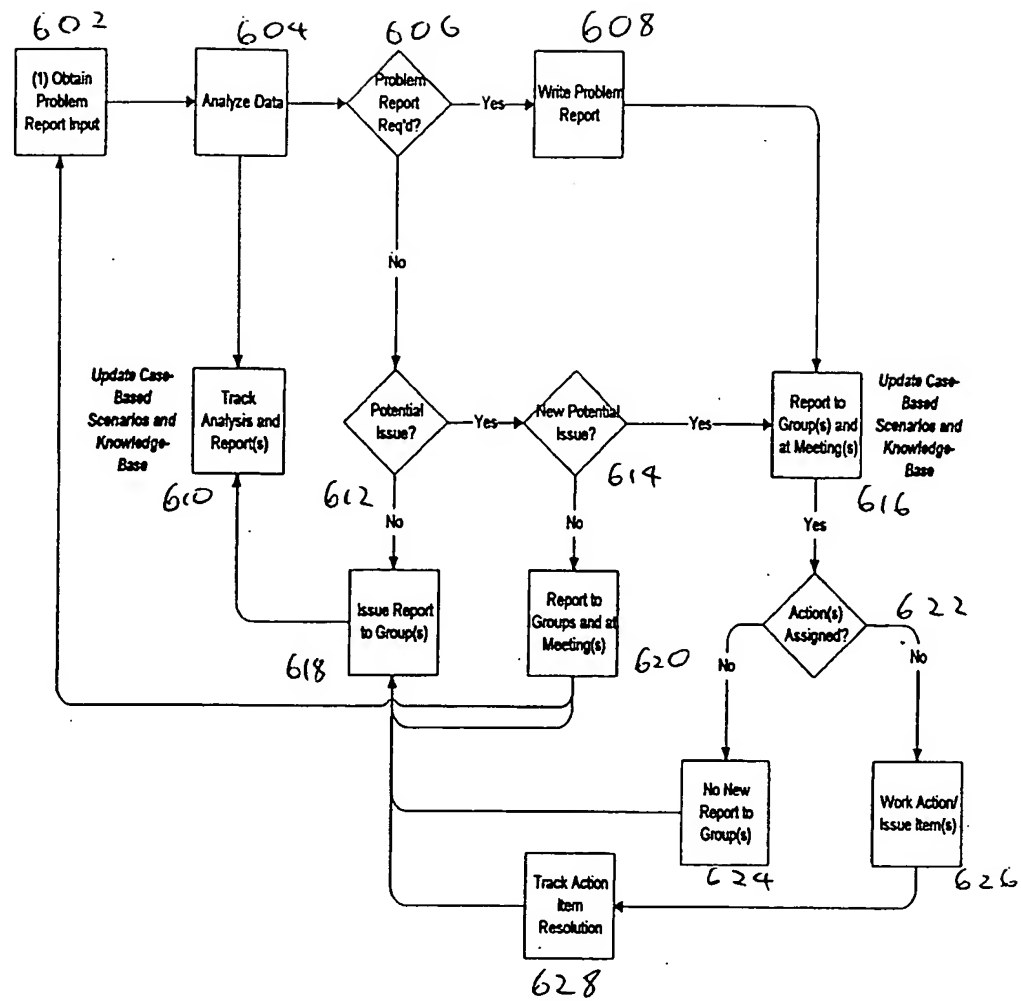


Figure 6

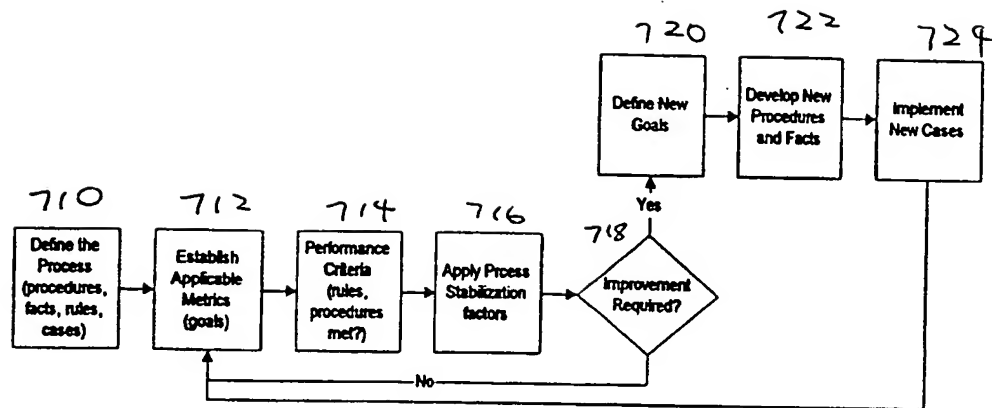
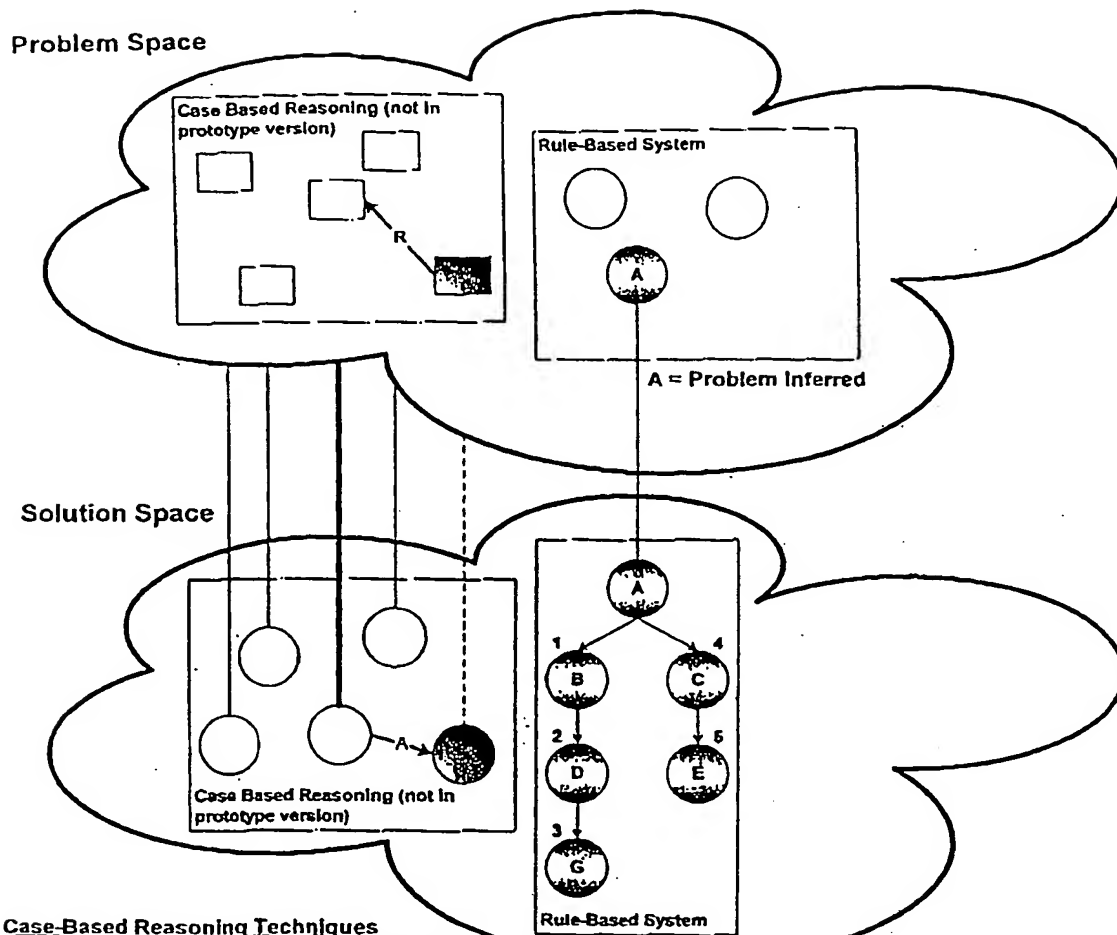






Figure 7

Figure 8



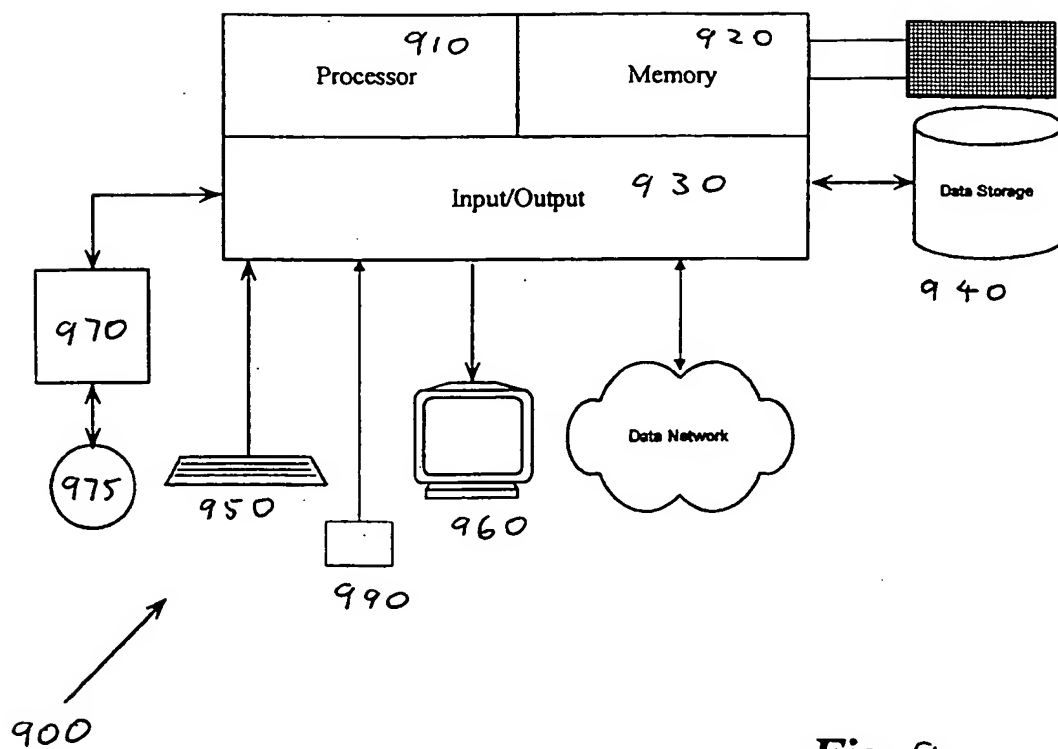
Case-Based Reasoning Techniques

A case is a contextualized piece of knowledge representing an experience. It contains the past lesson that is the content of the case and the context in which the lesson can be used. In the iCAN system, the problem space describes the state of the project or scenario when the case occurred, and the solution space states the derived solution to that problem. In the figure above, the description of a new problem to be solved is positioned in the problem space. Retrieval identifies the case with the most similar problem description. If necessary, adaptation occurs and a new solution is created. The iCAN system will use indexing to speed retrieval of data.

-  = Description of new problem to solve
-  = Description of solved problems
-  = Stored solutions
-  = New solution created by adaptation

Rule-Based Expert System

In Rule-based systems, knowledge is represented as facts about the world (i.e., relationships between entities) and rules for manipulating the facts. This apparent simplicity is complicated by the problems that at any one time more than one rule may apply and that as each rule is applied many more may become applicable. The iCAN system will implement a control structure to decide which rule to apply next and how to chain rules together. The iCAN system will use a "Depth-First Forward Chaining" control structure as shown in the above diagram. An inherent problem in inference engines is in which order to process rules - a decision must be made as to which rule to "fire" first. In the Depth-First Forward Chaining control structure, inferences would be made based on the order indicated as shown in the figure above. Thus, in the iCAN system, G would be inferred as true before it is inferred that E was true. If G is a solution, there may not be a need to infer that E was also a potential solution.

*Fig. 9*